УДК 519.6

ПРИМЕНЕНИЕ ГРАФИЧЕСКИХ АРИФМЕТИЧЕСКИХ УСКОРИТЕЛЕЙ НА МЕТОДИЧЕСКОЙ ПРОГРАММЕ РЕШЕНИЯ УРАВНЕНИЯ ПЕРЕНОСА МЕТОДОМ МОНТЕ-КАРЛО

И. А. Крючков, С. П. Огнев, А. С. Рыбкин, С. А. Степаненко, В. В. Южаков
(РФЯЦ-ВНИИЭФ)

Проведено исследование целесообразности применения вычислительной системы с гибридной архитектурой, содержащей MIMD- и SIMD-компоненты, для ускорения методической программы решения уравнения переноса методом Монте-Карло.

Результаты тестирования приведены для вариантов систем, содержащих один и два арифметических ускорителя, а также варианта одновременного использования универсального процессора и арифметического ускорителя.

Ключевые слова: графический арифметический ускоритель, GPU, процессор, уравнение переноса, метод Монте-Карло.

Введение

Метод Монте-Карло — один из наиболее распространенных методов решения уравнения переноса нейтронов и гамма-квантов, так как позволяет с наименьшими упрощениями с точки зрения как геометрии, так и константного обеспечения моделировать эти процессы [1—3].

Появление и развитие многопроцессорных вычислительных комплексов также способствовало укреплению позиций метода в решении уравнения переноса. Распараллеливание по траекториям частиц для задач линейного переноса позволяет получать высокую эффективность на большом числе процессоров [4]. Несколько ниже эффективность у алгоритмов для распараллеливания решения задач на собственные значения [4].

Целью данной работы является исследование целесообразности применения вычислительной системы с гибридной архитектурой для ускорения решения задач на собственные значения в рамках решения уравнения переноса методом Монте-Карло.

Вычислительная система содержит вычислительный модуль, являющийся MIMD-компонентом, и один или два графических арифметических ускорителя (АрУ), представляющих собой SIMD-компонент.

Результаты, которые приводятся в данной работе, получены на методической программе расчета временной постоянной размножения нейтронов методом установления. Данная программа близка по своей сути и к программам расчета энерговыделения.

Описание методической программы

При решении уравнения переноса методом Монте-Карло происходит моделирование траекторий частиц в системе и вычисление на них функционалов. В виде функционалов в методе Монте-Карло представляются различные физические характеристики, например потоки частиц или энергии, числа реакций и т. п., которые являются результатом расчета. Таким образом, расчет физических величин сводится к моделированию независимых траекторий частиц и оценке на них необходимых функционалов.

Окончание моделирования траектории происходит по одной из следующих причин:

- частица вылетает из системы;
- частица поглощается;
- частица гибнет по условию, заданному в тактике счета задачи.

Расчет временной постоянной размножения нейтронов методом установления осуществляется с заданным шагом по времени, поэтому появляется дополнительное условие прекращения моде-

лирования траектории частицы в пакете — достижение заданного момента времени.

Для такого класса задач траектория нейтронов уже не является достаточно независимой в том смысле, что, закончив моделироваться в заданный момент времени в текущем пакете, она продолжает моделироваться в следующем пакете. Поэтому каждый последующий пакет представляет собой ансамбль очередного поколения нейтронов по временным шагам. Это обстоятельство делает необходимым сохранение фазовых параметров ансамбля нейтронов (пространственных координат, направляющих косинусов вектора скорости, энергии и т. д.) в конце расчета траекторий одного пакета (массив конечных поколений) и передачу их для начала расчета траекторий следующего пакета (массив начальных поколений).

Для решения рассматриваемого класса задач на многопроцессорных ЭВМ используется распараллеливание по траекториям частиц. Распараллеливание производится в пределах текущего пакета (ансамбля) траекторий частиц. Все траектории в пакете распределяются между процессорами так, что каждый процессор моделирует свою группу из одной или более траекторий, в зависимости от размера пакета и числа процессоров, заказанных для расчета.

Структура системы с гибридной архитектурой

Структура гибридной вычислительной системы представлена на рис. 1. Она содержит универсальный процессор семейства Kentsfield Intel Core 2 Quad Q6600, работающий на тактовой частоте 2,4 ГГц, и два АрУ, подключенных с использованием шины PCI-Express 1.1 x16.

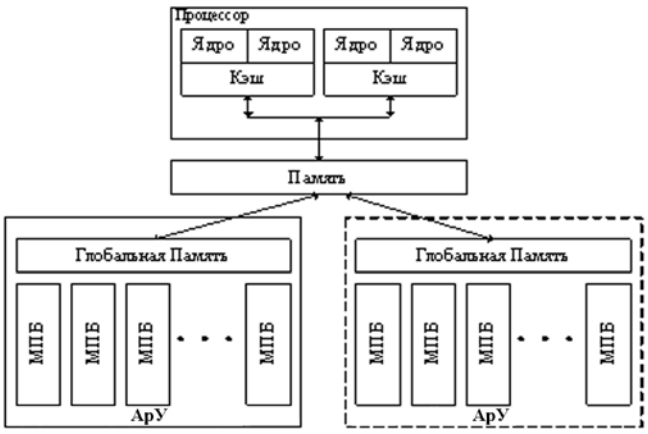

Рис. 1. Структура вычислительного модуля

АрУ состоит из процессора NVIDIA G80 и оперативной памяти типа GDDR3 объемом 768 Мб. Схема АрУ представлена на рис. 2,*а*.

Процессор G80 содержит 16 мультипроцессорных блоков (МПБ) и 6 контроллеров памяти (КП), обеспечивающих подключение 64-битных линий к оперативной памяти. Каждый МПБ включает 8 потоковых процессоров (ПП), массив из 8192 32-битных регистров и общую память объемом 16 Кб (рис. 2,*б*).

Глобальная, локальная, текстурная память, а также память констант располагаются в оперативной памяти АрУ (рис. 3). Память констант и текстурная память являются кэшируемыми. Объем каждой составляет 8 Кб на каждый МПБ.

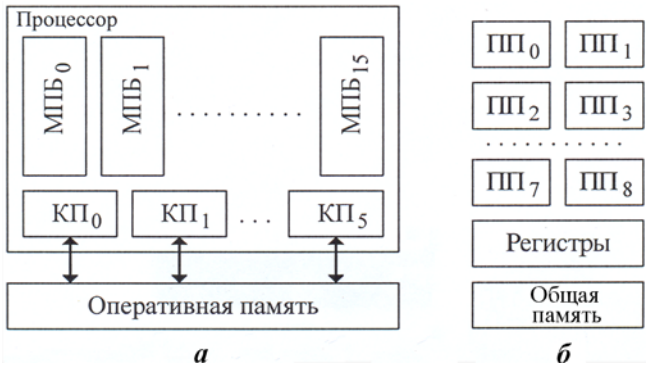

Рис. 2. Схемы АрУ (*а*) и МПБ (*б*)

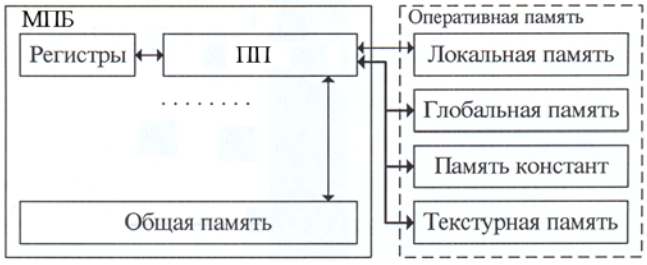

Рис. 3. Схема взаимодействия МПБ с оперативной памятью АрУ

Основные определения

Данные для исполнения поступают в АрУ в виде потоков (блоков потоков).

Каждый блок потоков состоит из Warp-ов. Warp — SIMD-группа потоков фиксированного

размера, состоящая из скалярных потоков с последовательными координатами.

Блок потоков всегда исполняется только на одном МПБ. Разделяемые переменные блока потоков хранятся в общей памяти МПБ.

МПБ может обрабатывать несколько блоков потоков одновременно. Локальный регистровый файл и общая память делятся между всеми потоками (блоками потоков). Тем самым с уменьшением числа используемых регистров (на поток) и размера используемой общей памяти (на блок) увеличивается число потоков (блоков потоков), способных одновременно находиться в обработке. На логическом уровне блоки потоков всегда изолированы.

Оценка использования АрУ

Чтобы исследовать целесообразность применения вычислительной системы с гибридной архитектурой для ускорения методической программы решения уравнения переноса методом Монте-Карло, требуется оценить загрузку МПБ АрУ.

Как показано в [5], для оценки использования АрУ необходимо сформировать последовательность инструкций, сконфигурировать устройство и передать поток входных данных. По предложению разработчиков программного обеспечения CUDA SDK это реализуется следующим образом: пользователь описывает ядро программы, т. е. набор процедур, которые будут исполняться на вычислительных блоках АрУ, и при запуске ядра указывает конфигурацию необходимых программных и аппаратных средств.

Основным моментом адаптации вычислительного алгоритма для АрУ является определение соответствующего ядра программы. Каждый поток, физически выполняющийся на арифметически-логическом устройстве МПБ, исполняет инструкции, описанные пользователем в ядре. При этом на каждом МПБ параллельно выполняются несколько потоков с одной и той же последовательностью инструкций. Следует учитывать, что каждый из потоков, исполняющих код ядра, для получения входных данных обращается к области памяти устройства. Логически потоки объединяются в блоки, ограничивающие возможность обмена данными между потоками. Потоки внутри одного блока могут обмениваться данными через общую память. Кроме того, гарантируется, что потоки из одного блока выполняются на одном и том же МПБ.

В табл. 1 приведены предельно допустимые параметры АрУ. Для выполнения на ускорителе адаптированного основного ядра программы в один поток требуются параметры, представленные в табл. 2.

Максимально достижимый коэффициент использования АрУ на задаче согласно [5] определяется следующим образом:

$$K^{\text{АрУ}} = \frac{W_a^{\text{МПБ}}}{W_{lim}^{\text{МПБ}}}.$$

Здесь $W_a^{\text{МПБ}}$ и $W_{lim}^{\text{МПБ}}$ — соответственно число активных и максимально возможное число Warp-ов в МПБ. В свою очередь,

$$W_a^{\text{МПБ}} = B_a^{\text{МПБ}} W;$$

$$B_a^{\text{МПБ}} = \min\left(X_{limW}^{\text{МПБ}}, X_{limS}^{\text{МПБ}}, X_{limR}^{\text{МПБ}}\right);$$

$$X_{limW}^{\text{МПБ}} = \min\left(N_b^{\text{МПБ}}, \left[\frac{W_{lim}^{\text{МПБ}}}{W^{\text{МПБ}}}\right]\right);^{*}$$

$$X_{limS}^{\text{МПБ}} = \left[\frac{S_{lim}^{\text{МПБ}}}{S^{\text{МПБ}}}\right]; \quad X_{limR}^{\text{МПБ}} = \left[\frac{R_{lim}^{\text{МПБ}}}{R^{\text{МПБ}}}\right];$$

*Как обычно, $[a]$ означает ближнее целое меньше a и соответственно $]a[$ — ближнее целое больше a.

Таблица 1

Предельно допустимые параметры АрУ

Параметр	Значение
Количество МПБ	16
Количество потоков в Warp-е ($C_{\text{п}}^W$)	32
Количество Warp-ов в МПБ ($W_{lim}^{\text{МПБ}}$)	24
Количество потоков в МПБ	768
Количество блоков потоков в МПБ	8
Общее количество 32-битных регистров в МПБ ($R_{lim}^{\text{МПБ}}$)	8 192
Емкость общей памяти в МПБ ($S_{lim}^{\text{МПБ}}$), байт	16 384

Таблица 2

Требуемые параметры

Параметр	Значение
Емкость локальной памяти на поток, байт	160
Емкость общей памяти на блок (S), байт	4 672
Количество 32-битных регистров на поток (R)	30

$$W^{\text{МПБ}} = \left\rceil \frac{C^{\text{МПБ}}}{C_{\text{п}}^{W}} \right\lceil;$$

$$S^{\text{МПБ}} = 512 \left\rceil \frac{S}{512} \right\lceil; \quad R^{\text{МПБ}} = 4 \left\rceil \frac{2W^{\text{МПБ}}}{4} \right\lceil 16R,$$

где $B_a^{\text{МПБ}}$ — максимальное число активных блоков потоков в МПБ; W — число Warp-ов в блоке; $X_{limW}^{\text{МПБ}}$, $X_{limS}^{\text{МПБ}}$, $X_{limR}^{\text{МПБ}}$ — ограничения соответственно по максимальному числу Warp-ов, по общей памяти и по числу регистров; $C^{\text{МПБ}}$ — число потоков в МПБ; остальные обозначения указаны в табл. 1, 2.

В соответствии с необходимыми ресурсами можно построить зависимости эффективности использования МПБ от числа потоков в нем и от количества регистров, используемых потоком (рис. 4).

Наиболее оптимальным числом потоков в МПБ является 128 при числе блоков потоков в МПБ, равном 2.

Главным ограничивающим фактором является число регистров, необходимое каждому потоку. При использовании меньшего, чем необходимо, количества регистров (см. табл. 2) компилятор размещает переменные в локальной памяти, доступ к которой неэффективен.

Исходя из требуемых параметров основного ядра программы, определим коэффициент использования АрУ:

$$K^{\text{АрУ}} = 0{,}33.$$

Оценка теоретического ускорения

Анализ разработанного адаптированного кода показал, что он содержит несколько инструкций, позволяющих выполнять суммирование и умножение за один такт. Доля таких инструкций в общем коде программы незначительна; число инструкций, выполняемых за такт, равно

$$I_{fpu}^{\text{АрУ}} \approx 1.$$

Следовательно, можно утверждать, что для данной реализации максимально достижимая теоретическая производительность арифметического ускорителя равна

$$P_{\max}^{\text{АрУ}} = H^{\text{АрУ}} N_{\text{АЛУ}}^{\text{АрУ}},$$

где $H^{\text{АрУ}} = 1{,}35\,\text{ГГц}$ — частота работы АрУ; $N_{\text{АЛУ}}^{\text{АрУ}} = 128$ — количество арифметико-логических устройств в АрУ. Получаем $P_{\max}^{\text{АрУ}} = 172{,}8\,\text{Гфлоп}$.

Таким образом, теоретически достижимое значение производительности ускорителя на этой программе

$$P_{\text{д}}^{\text{АрУ}} = K^{\text{АрУ}} P_{\max}^{\text{АрУ}} = 58\,\text{Гфлоп},$$

где $K^{\text{АрУ}} = 0{,}33$.

Теоретическое пиковое значение производительности ядра универсального процессора $P^{\text{ЦП}} = 4{,}8\,\text{Гфлоп}$ (в силу того, что код не использует инструкции SSE). Ожидаемое значение ускорения выполнения исполняемого кода программы равно

$$k = \frac{P_{\text{д}}^{\text{АрУ}}}{P_{\text{д}}^{\text{ЦП}}} = \frac{K^{\text{АрУ}} P_{\max}^{\text{АрУ}}}{P^{\text{ЦП}}} = \frac{0{,}33 \cdot 172{,}8}{4{,}8} \approx 12.$$

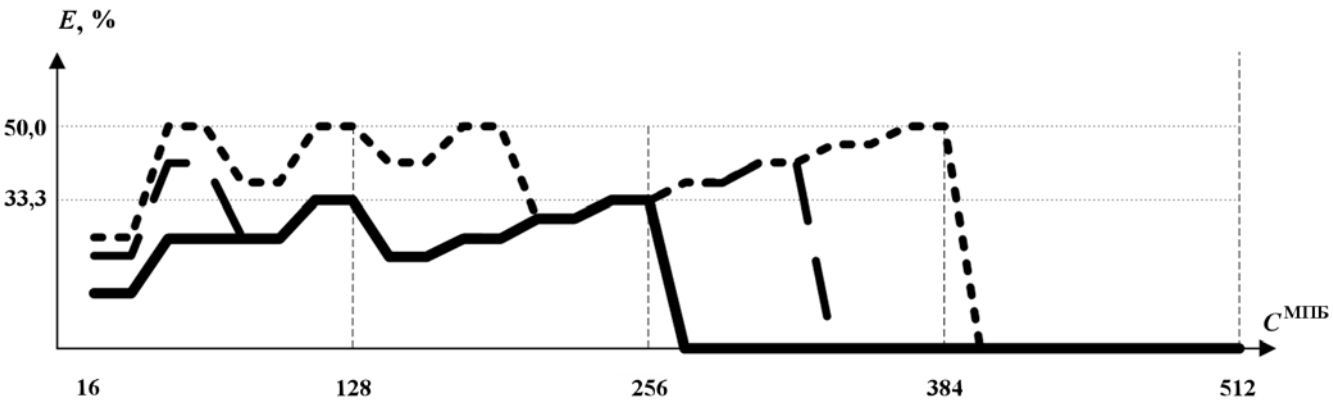

Рис. 4. Эффективность использования мультипроцессора: – – – — до 21 регистра; — — — от 22 до 25 регистров; ——— — от 26 до 32 регистров

Результаты тестирования

Результаты тестирования приведены для двух различных вариантов вычислителей:

– с использованием одного АрУ (вариант 1);

– с использованием двух АрУ (вариант 2).

Вариант 1. **Вычислительная система с одним ускорителем.** Схема выполнения программы при использовании одного АрУ представлена на рис. 5. Программа состоит из трех больших блоков, разделенных командами MPI_Barrier. В первом верхнем блоке происходит счет на универсальном процессоре всей необходимой информации. Во втором блоке этот же счет выполняется на АрУ. В третьем происходит сравнивание результатов, полученных на универсальном процессоре и на АрУ.

Общая длительность выполнения программы $T_{\text{общее}}$ состоит из интервалов, обозначенных на рис. 5 через T_i $(i = \overline{0,8})$:

$$T_{\text{общее}} = \sum_{i=0}^{8} T_i.$$

При этом длительность выполнения процедур на универсальном процессоре равна

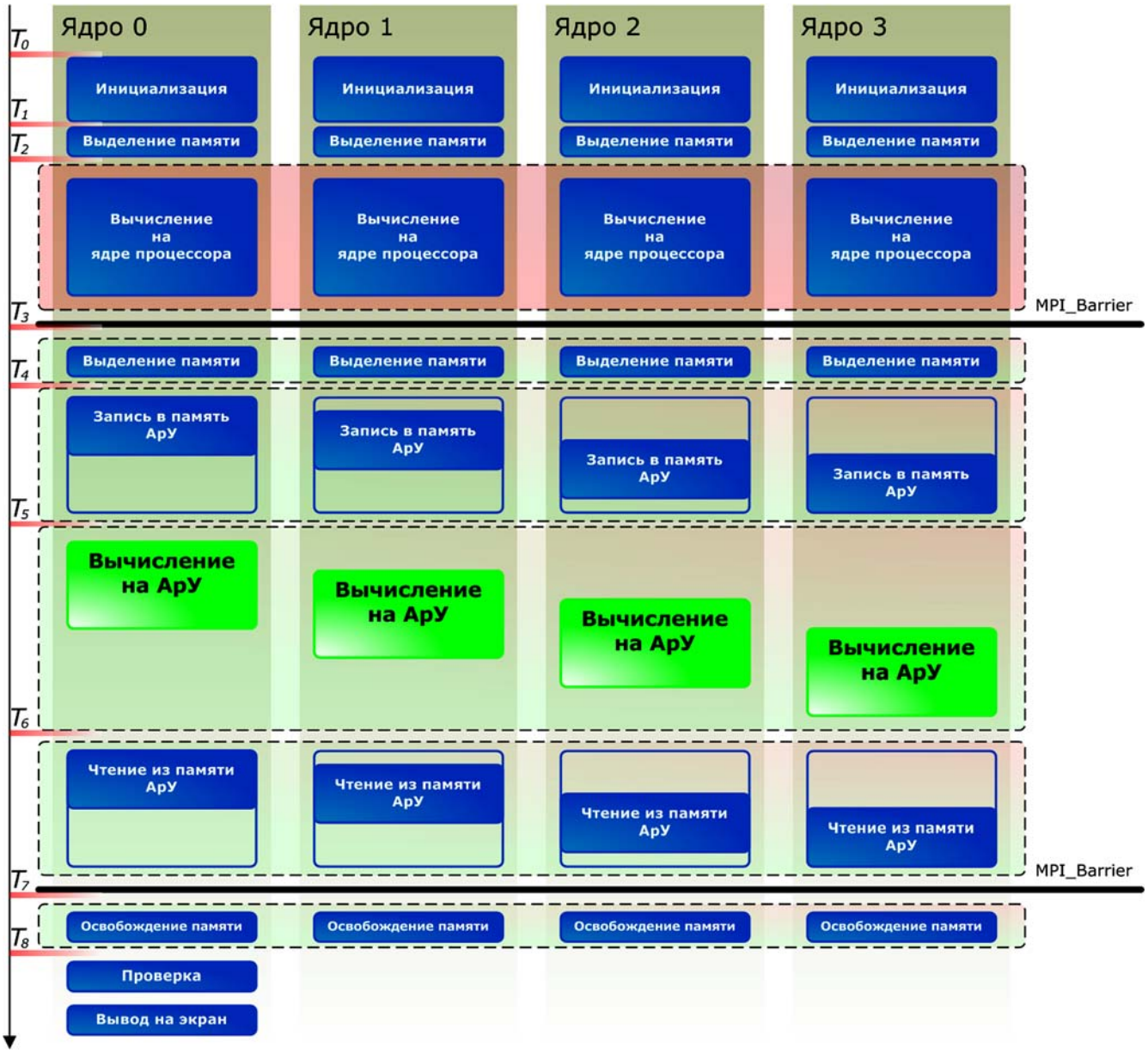

Рис. 5. Схема выполнения программы (вариант 1)

$$T_j^{\text{ЦП}} = T_3 - T_2,$$

где j — количество задействованных ядер универсального процессора. Отметим, что $T_j^{\text{ЦП}} \gg \gg T_2 - T_0$ и, следовательно, указанный интервал можно не учитывать.

Длительность выполнения на АрУ равна

$$T_j^{\text{АрУ}} = T_8 - T_3.$$

Коэффициент ускорения программы

$$k_j = \frac{T_j^{\text{ЦП}}}{T_j^{\text{АрУ}}}.$$

Выполнение кода на АрУ осуществлялось с использованием NVIDIA CUDA 1.1. На универсальном процессоре использовался компилятор Microsoft Visual C++ 2005. Распараллеливание осуществлялось с использованием интерфейса MPI. Для компилятора выставлена максимальная высокоуровневая оптимизация. Вычисления выполнялись с одинарной точностью.

Ускорения выполнения программы представлены на рис. 6. Распараллеливание на каждом шаге осуществлялось по числу траекторий частиц M.

В результате применения АрУ длительность всего вычислительного процесса на гибридном вычислителе уменьшена в 12,3 раза по сравнению с применением одного ядра универсального процессора. Ускорение по сравнению с двумя и четырьмя ядрами процессора составило 6,3 и 3,2 соответственно.

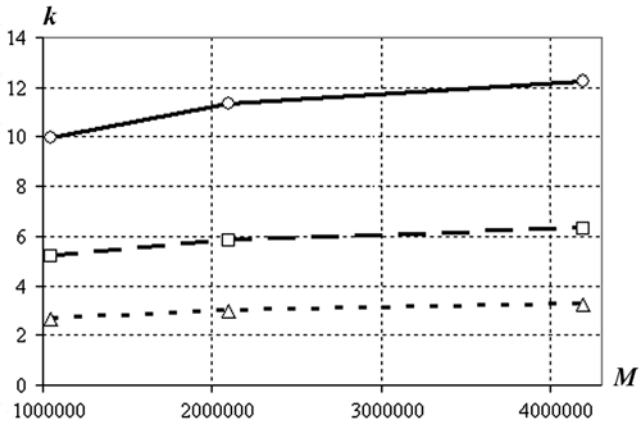

Рис. 6. Ускорение выполнения программы на гибридной системе по сравнению с выполнением на ядрах процессора: —○— на 1 ядре; — □ — на 2 ядрах; - - △ - - на 4 ядрах

Вариант 2. **Вычислительная система с двумя ускорителями.** Принципиальным отличием от варианта 1 является уменьшение $T_{\text{счет}}^{\text{АрУ}} = T_6 - T_5$ за счет использования двух АрУ.

Длительность всего вычислительного процесса при использовании двух АрУ на гибридном вычислителе уменьшена в 6,2 раза по сравнению с использованием четырех ядер универсального процессора.

Заключение

Адаптация программы и применение вычислительной системы с гибридной архитектурой, содержащей MIMD-компонент — универсальный процессор и SIMD-компонент — АрУ, позволили выполнить тестовые расчеты в двух следующих вариантах:

1. Вариант с использованием в составе системы одного АрУ. Длительность выполнения расчетов уменьшилась в 12,3 раза по сравнению с использованием одного ядра универсального процессора. Ускорение по сравнению с выполнением на двух и четырех ядрах составило 6,3 и 3,2 соответственно.
2. Вариант с одновременным задействованием двух АрУ. При использовании в составе системы двух АрУ длительность выполнения расчетов уменьшена в 6,2 раза по сравнению с использованием четырех ядер универсального процессора; т. е. ускорение здесь практически пропорционально количеству ускорителей.

Полученные значения ускорения хорошо совпадают с теоретическими оценками на основе пиковых производительностей процессора и ускорителя.

Ускорение вычислительного процесса может быть увеличено в результате применения ускорителей с большим количеством 32-битных регистров в МПБ.

В дальнейшем целесообразно оптимизировать работу с глобальной памятью ускорителя.

Полученные результаты показывают перспективность применения MIMD- и SIMD-компонентов (в частности, не только NVIDIA GeForce 8800GTX, но и AMD FireStream 9170, IBM PowerXCell8i и других перспективно разрабатываемых гибридных процессоров) для решения уравнений переноса методом Монте-Карло.

Список литературы

1. *Спанье Дж., Гелбард Э.* Метод Монте-Карло и задачи переноса нейтронов. М.: Атомиздат, 1972.
2. *Соболь И. М.* Численные методы Монте-Карло. М.: Наука, 1973.
3. *Субботин А. Н., Ченцов Н. Н.* Моделирование процесса рассеяния частиц в методе Монте-Карло // Вопросы атомной науки и техники. Сер. Методики и программы численного решения задач математической физики. 1982. Вып. 1. С. 3—9.
4. *Житник А. К., Огнев С. П., Иванов А. Н.* Параллельные алгоритмы для решения методом Монте-Карло задач переноса и задач на собственные значения на многопроцессорных вычислительных комплексах с распределенной памятью // Труды РФЯЦ-ВНИИЭФ. 2002. Вып. 3. С. 76—87.
5. NVIDIA CUDA Compute Unified Device Architecture Programming Guide. Ver. 1.1. November 2007. NVIDIA Corporation, 2007.

Статья поступила в редакцию 16.10.08.